Sheet 4 (2D graphics)

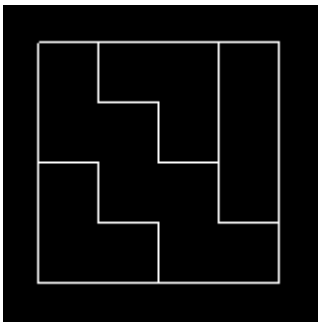
1. The following OpenGL program displays a sphere by approximating it in terms of rectangles and triangles.
    a) Write and execute the program. Does your program output appear as a sphere? If not explain why.
    b) Change the step taken along both theta and phi angles to 5, 10, 25 degrees and rerun your program. What you note? Explain.

```
#include "stdafx.h"
#include <GL/glut.h>
#include <math.h>
void mydisplay()
{
    GLdouble c=3.1444/180.0;
    GLdouble phi80r=c*80.0;
    GLdouble x,y,z,theta,thetar,phi,phir,phir20;
    glClear(GL_COLOR_BUFFER_BIT);
    // north pole
    glBegin(GL_TRIANGLE_FAN);
    glVertex3d(0.0, 0.0 , 1.0);
    z=sin(phi80r);
    for(theta=-180.0; theta<=180.0;theta+=20.0)
    {
        thetar=c*theta;
        x=sin(thetar)*cos(phi80r);
        y=cos(thetar)*cos(phi80r);
        glVertex3d(x,y,z);
    }
    glEnd();
    // south pole
    glBegin(GL_TRIANGLE_FAN);
    glVertex3d(0.0, 0.0 , -1.0);
    z=-sin(phi80r);
    for(theta=-180.0;theta<=180.0;theta+=20.0)
    {
        thetar=c*theta;
        x=sin(thetar)*cos(phi80r);
        y=cos(thetar)*cos(phi80r);
        glVertex3d(x,y,z);
    }
    glEnd();
    // middle
    for(phi=-80.0; phi<=80.0; phi+=20.0)
    {
        phir=c*phi;
        phir20=c*(phi+20);
        glBegin(GL_QUAD_STRIP);
        for(GLfloat theta=-180.0; theta<=180.0;theta+=20.0)
        {
            thetar=c*theta;
            x=sin(thetar)*cos(phir);
            y=cos(thetar)*cos(phir);
            GLfloat z=sin(phir);
            glVertex3d(x,y,z);
            x=sin(thetar)*cos(phir20);
            y=cos(thetar)*cos(phir20);
            z=sin(phir20);
            glVertex3d(x,y,z);
        }
        glEnd();
    }
    glFlush();
}
int main(int argc, char** argv){
    glutCreateWindow("Sphere");
    glutDisplayFunc(mydisplay);
    glutMainLoop();
}
```

2. Describe how you would adapt the RGB-color model in OpenGL to allow you to work with a subtractive color model (Problem 2.8).

3. In OpenGL, we can associate a color with each vertex. If the endpoints of a line segment have different colors assigned to them, OpenGL will interpolate between the colors as it renders the line segment. It will do the same for polygons. Use this property to display the Maxwell triangle: an equilateral triangle whose vertices are red, green, and blue (Problem 2.15).

4. Write an OpenGL program to generate a rectangular array of cells. Each cell has four sides. Each cell, except cells on the boundary, must be open from at least one edge, see the figure below. Your program should be able to generate any grid of any given number of rows/columns.



5. (As project) Write an OpenGL program that, starting with the rectangular array you created in the last question, Creates a maze. To do that, you remove existing cell sides (except from the perimeter of all the cells) until all the cells are connected. Then you create an entrance and an exit for the maze anywhere on the boundary; see the figure below (Problem 2.7).